Presented at 3rd IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2003)
8-10 Sep 2003, Benalmádena, Spain

A new point cloud simplification algorithm

Carsten Moenning
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD - UK
email: cm230@cl.cam.ac.uk

Neil A. Dodgson
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD - UK
email: nad@cl.cam.ac.uk

ABSTRACT
We present a new technique for the simplification of point-sampled geometry without any prior surface reconstruction. Using Fast Marching farthest point sampling for implicit surfaces and point clouds [1], we devise a coarse-to-fine uniform or feature-sensitive simplification algorithm with user-controlled density guarantee. The algorithm is computationally and memory efficient, easy to implement and inherently allows for the generation of progressive and multiresolution representations of the input point set.

KEY WORDS
Point cloud simplification, farthest point sampling, Fast Marching.

1 Introduction

Due to recent advances in surface acquisition techniques, 3D object boundary surfaces are now commonly acquired with submillimeter accuracy. The initial output of acquisition devices such as laser range scanners therefore generally consists of point clouds of considerable redundancy. As part of a typical mesh processing pipeline, these point sets are converted into polygonal mesh representations of substantial size with the help of often computation and memory demanding surface reconstruction algorithms. The size of the resulting meshes frequently makes any further processing without prior and often costly mesh simplification impossible.

Point cloud simplification represents an attractive alternative to this process. By simplifying the point set first, any subsequent surface reconstruction becomes significantly faster and mesh simplification becomes obsolete. Point cloud simplification tends to be computationally more efficient and less memory demanding than mesh simplification since no mesh data structures need to be maintained. Furthermore, with the increasing availability of powerful point-based modelling [2], multiresolution [3] and visualisation [4, 5] techniques, the simplification of dense point clouds for subsequent point-based rather than polygonal mesh-based processing is of significant interest by itself.

Subject to a user-controlled minimum density condition, we therefore consider the problem of simplifying a densely or non-uniformly distributed unstructured point cloud, $P = \{p_1, p_2, \ldots, p_{N_1}\}$, to a target model size $N_2 < N_1$. The point set is assumed to represent the surface of a smooth, two-manifold boundary of a 3D model.

1.1 Previous Work

Dey et al. [6] present a point cloud simplification algorithm with user-controlled density guarantee which detects redundancy in the input point cloud with the help of the "Cocone" and local feature size [7] concepts. The decimation is inherently sensitive to changes in local curvature and supports high-quality reconstructions. The algorithm does not allow for adaptive decimation driven by changes in a measure other than or in addition to local curvature. The method requires the computation and maintenance of 3D Voronoi diagrams and tends to be computationally and memory expensive.

Boissonnat and Cazals [8] introduce a coarse-to-fine point cloud simplification approach. Their algorithm takes a random initial subset of the input point cloud and uses its 3D Delaunay triangulation to define a signed distance function over the set. This implicit function is then used to enlarge the initial set until a significant number of points is found to lie within a user-defined approximation error tolerance. In the second and final step, the enlarged subset is Delaunay-triangulated [12] and a surface mesh is reconstructed. If this initial surface does not meet the error condition, additional points are inserted iteratively sorted by their distance to the closest surface facet. The method thus delivers both a reconstructed and simplified mesh simultaneously. The algorithm's point cloud simplification step is costly due to the construction and maintenance of a 3D Delaunay triangulation. Given the increasing usefulness of point-based processing, the algorithm's restriction to triangular meshes as output seems undesirable.

Linsen [2] associates an information content measure with every input point and subsequently removes points featuring the lowest entropy. His information measure allows for local curvature and RGB colour changes. The algorithm is simple and produces visually appealing results but gives no guarantees on the density of the output point set. Extremely non-uniformly distributed input point sets will therefore necessarily result in simplified point sets of insufficient density. Similarly, even highly dense input point clouds may be simplified to prohibitively unevenly distributed output point sets. In either case, resampling of the input cloud may be necessary to support any effective further processing.

In an important paper, Pauly et al. [9] adapt various widely used mesh simplification techniques to the point cloud simplification scenario. Their quadric error-based iterative simplification method produces point sets with low approximation error but is very sensitive in execution time to the size of the input point set. The authors' particle simulation technique also results in point sets of low average error but is generally relatively inefficient to compute. Pauly et al. [9] uniform incremental clustering method is computationally efficient at the expense of a relatively high average approximation error and is not naturally extensible to adaptive, feature-sensitive simplification. Their hierarchical clustering algorithm is execution time and memory efficient but in its feature-sensitive version produces point sets of approximation error only slightly lower than that introduced by uniform incremental clustering.

Alexa et al. [5] uniformly reduce point cloud redundancy by estimating a point's contribution to the moving least squares (MLS) representation of the underlying surface. Those points contributing the least are subsequently removed. Similar to Linsen's [2] algorithm, this method does not guarantee the absence of insufficiently dense output point sets. Alexa et al. [5] therefore suggest a resampling method which computes planar Voronoi diagram representations of undersampled MLS surface regions. The density of the point set is increased by iteratively choosing the vertex of the Voronoi diagram which is farthest away from any of the other surface points in the diagram. This process is repeated until the Euclidean distance between the next sampling candidate and its nearest point is less than a user-specified threshold.

1.2 Contribution of this paper

Using our Fast Marching farthest point sampling method for implicit surfaces and point clouds (FastFPS) [1], we present a new coarse-to-fine point cloud simplification algorithm without the need for any prior surface reconstruction. The algorithm is execution time and memory efficient in both its uniform and adaptive, feature-sensitive form. The user-controlled density of the output point set is guaranteed thereby guaranteeing the availability of a point set sufficiently dense for meaningful further processing. In its uniform version, the method produces irregular cluster- and hole-free point sets exhibiting excellent anti-aliasing properties. The method allows for feature-sensitive simplification in the form of any combination of point weights such as local surface variation, colour difference estimates, etc. either computed on-the-fly or imported in the form of pre-computed importance maps. The coarse-to-fine nature of the algorithm inherently supports the generation of progressive and multiresolution representations of the input point cloud.

Our algorithm is closest in nature to Alexa et al. [5] resampling method. In contrast to their work, our simplification method is coarse-to-fine throughout and does not require any resampling to guarantee the user-requested point set density. Furthermore, our algorithm supports both uniform and feature-sensitive point cloud simplification. The Voronoi diagrams computed by our method represent true (discrete) surface Voronoi diagrams rather than local, planar approximations. As a result, the point sets returned by the algorithm's uniform version are truly irregularly uniform in a deterministic sense and retain the excellent space-filling and anti-aliasing properties typical of farthest point sequences [11].

2 FastFPS point cloud simplification

In the following, we introduce our coarse-to-fine point cloud simplification algorithm. We start by briefly summarising the underlying FastFPS technique [1], followed by the presentation of the simplification algorithm itself.

2.1 Fast Marching farthest point sampling for implicit surfaces and point clouds

Farthest point sampling [10, 11] is intuitively based on the idea of minimising any reconstruction error by repeatedly placing the next sample point in the middle of the least-known area of the sampling domain. Eldar et al. [11] show that in the case of a farthest point sequence the location of the next point to be sampled coincides with a vertex of the bounded Voronoi diagram [12] of the previously selected set of samples S, $BVD(S)$. Thus, incremental Voronoi diagram construction provides farthest point samples progressively. To farthest point-sample a 3D point cloud, our FastFPS algorithm therefore computes (discrete) Voronoi diagrams in the form of weighted distance maps incrementally directly across the input point set. This is achieved efficiently and without the need for any prior surface reconstruction by using Mémoli and Sapiro's [13, 14] recent extension of the original Fast Marching level set method [15].

Mémoli and Sapiro's [13, 14] extended Fast Marching technique considers a closed hyper-surface M in $\mathbb{R}^m$ given as the zero level-set of a distance function $\phi : \mathbb{R}^m \to \mathbb{R}$, $m \geq 3$. The r-offset, Ω_r, of M is given by the union of the balls centred at the surface points with radius r

$$\Omega_r := \bigcup_{x \in M} B(x, r) = \{x \in \mathbb{R}^m : |\phi(x)| \leq r\} \qquad (1)$$

For a smooth M and r sufficiently small, Ω_r is a manifold with smooth boundary [13]. To compute the weighted distance map originating from a source point $q \in M$ and propagating with speed $F(p)$ on M, Mémoli and Sapiro [13] suggest using the Euclidean distance map in Ω_r to approximate the intrinsic distance map on M. That is

$$|\nabla_M T_M(p)| = F(p) \qquad (2)$$

for $p \in M$ and with boundary condition (propagation source point) $T_M(q) = 0$ is approximated by

$$|\nabla T_{\Omega_r}(p)| = \tilde{F}(p) \tag{3}$$

for $p \in \Omega_r$ and boundary condition $T_{\Omega_r}(q) = 0$. $\tilde{F}(p)$ represents the (smooth) extension of $F(p)$ on M into Ω_r. The problem of computing an intrinsic distance map has therefore been transformed into the problem of computing a extrinsic distance map in an Euclidean manifold with boundary. Mémoli and Sapiro [13, 14] subsequently show that the Fast Marching method can be used to approximate the solution to (3) in a computationally optimal manner by only slightly modifying the original Fast Marching technique [15] to deal with bounded spaces. Mémoli and Sapiro [13] prove that their algorithm is of $O(N \log N)$ complexity, N representing the number of grid points in Ω_r. For further details, see Mémoli and Sapiro [13].

2.2 A new point cloud simplification algorithm

For simplicity, we first consider the formulation of our uniform point cloud simplification strategy.

The algorithm proceeds with the embedding of the given point cloud $P = \{p_1, p_2, \ldots, p_{N_1}\}$ in a Cartesian grid sufficiently large as to allow for the thin offset band Ω_r. For details on the determination of the optimal size of r, see Moenning and Dodgson [1].

Given an initial subset of input points $S \subset P$ in Ω_r, we then construct $BVD(S)$ by simultaneously propagating fronts from each of the initial points outwards. During this propagation, only grid points in Ω_r are considered. This process is equivalent to the computation of the Euclidean distance map across P given S and Ω_r. It is achieved by solving (3) with $\tilde{F}(p) = 1$ and using a single min-heap.

The vertices of $BVD(S)$ are given by those grid points entered by three or more propagation waves (or two for points on the domain boundary) and are therefore obtained as a by-product of the propagation process. The Voronoi vertices' arrival times are inserted into a max-heap data structure. The algorithm then proceeds by extracting the root from the max-heap, the grid location of which represents the location of the next output point. The point is inserted into $BVD(S)$ by resetting its arrival time to zero and propagating a front away from it. The front will continue propagating until it hits grid points featuring lower arrival times and thus belonging to neighbouring Voronoi cells. The T_{Ω_r} values of updated grid points are updated correspondingly in the min-heap using back pointers. New and obsolete Voronoi vertices are inserted or removed from the max-heap respectively. The algorithm continues extracting the root from the max-heap until the user-defined density condition has been met or a given target model size has been reached.

The density condition is formulated in terms of the maximum distance, ρ, between points permitted by the user. Thus, the simplified point set is refined until the next farthest point candidate's distance map value is no longer larger than the user threshold. Figure 1 shows the effect of different values of ρ on the density of the simplified point set.

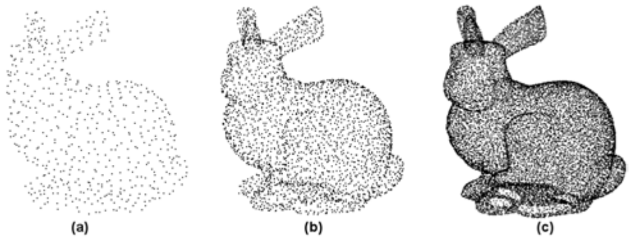

Figure 1. Effect of different user-controlled minimum density values, ρ, on density of the simplified 3D point set. (a) $\rho = 10$ (b) $\rho = 5$ (c) $\rho = 2.5$.

Since points are selected in Ω_r, the equivalent points directly on P are found at any given time by projecting the sample points in Ω_r onto the surface. The algorithm can thus be summarised as follows

0) Embed the given point cloud in a Cartesian grid sufficiently large to allow for an offset band of size r. Given an initial point cloud subset $S \in \Omega_r$, $n = |S| \geq 2$, compute $BVD(S)$ by propagating fronts with speed $\tilde{F}(p_i)$ from the points outwards using extended Fast Marching. Store the Voronoi vertices' arrival times in a max-heap.

1) Extract the root from the max-heap to obtain s_{n+1}. $S' = S \cup \{s_{n+1}\}$. Compute $BVD(S')$ by propagating a front locally from s_{n+1} outwards using extended Fast Marching and a single min-heap.

2) Correct the arrival times of updated grid points in the min-heap. Insert the vertices of the bounded Voronoi cell of s_{n+1}, $BV(s_{n+1})$, in the max-heap. Remove obsolete Voronoi vertices of the neighbours of $BV(s_{n+1})$ from the max-heap.

3) If neither the user-controlled point density ρ nor the target model size $N_2 < N_1$ has been reached, loop from 1).

By allowing $\tilde{F}(p_i)$ to vary with any (positive) weight associated with points $p_i \in P$, this algorithm supports flexible feature-sensitive simplification. By weighting the p_i by, for example, local surface variation and/or colour difference estimates, points are concentrated in regions of change in curvature and/or colour. Any feature-sensitivity estimates may be computed on-the-fly during front propagation using local neighbourhoods which at any time are readily available from the bounded Voronoi diagram. Alternatively, point weights may be passed to the algorithm in the form of an importance map. To allow for the user-controlled density, the point cloud is simplified uniformly until the density requirement is met. The remaining target model budget is then distributed sensitively to local features.

As regards the complexity of this algorithm, extracting the root from and inserting into the max- or min-heap and removing from the max-heap with subsequent re-heapifying are $O(\log W)$ operations, where W represents the number of elements in the heap. W is $O(N)$, N representing the number of grid points in the offset band Ω_r. The updating of the arrival times of existing min-heap entries is $O(1)$ due to the use of back pointers from the grid to the heap. The detection of a (bounded) Voronoi cell's vertices and, if required, boundary is a by-product of the $O(N \log N)$ front propagation. Thus, the algorithm's asymptotic efficiency is $O(N \log N)$.

3 Worked examples and discussion

We start by considering the uniform simplification of the point clouds acquired from the surface of the genus-0 "Venus" model and a genus-1 CAD object respectively. For simplicity, the radius of the union of balls making up the thin offset band Ω_r is set to a constant $r = 2$. Figures 2 and 3 present both the simplified point sets produced by our algorithm for various levels of detail and the renderings of the corresponding reconstructed triangular meshes. The irregular uniformity of the simplified point sets is evident in the cluster- and hole-free coverage of the two surfaces. This results in excellent anti-aliasing properties as can be seen from the quality of the renderings for relatively strongly simplified point sets. This anti-aliasing property alongside the coarse-to-fine level-of-detail nature of the algorithm yields immediate support for applications such as progressive transmission of the 3D content across a limited bandwidth channel or the generation of multiresolution representations.

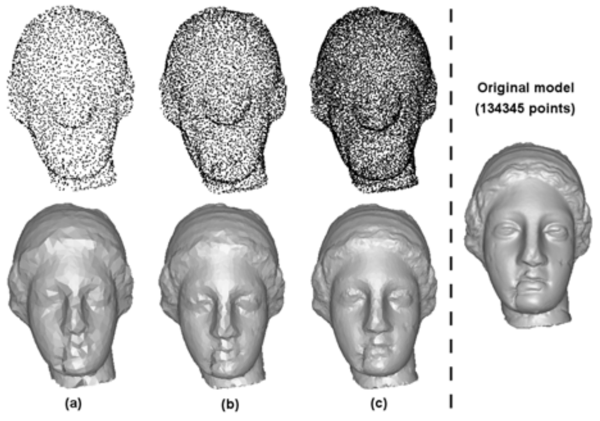

Figure 2. Uniformly simplified Venus point sets and the renderings of their mesh reconstructions.
(a) 97.5% simplified ($\rho = 5.00$).
(b) 95.0% simplified ($\rho = 2.50$).
(c) 90.0% simplified ($\rho = 1.25$).

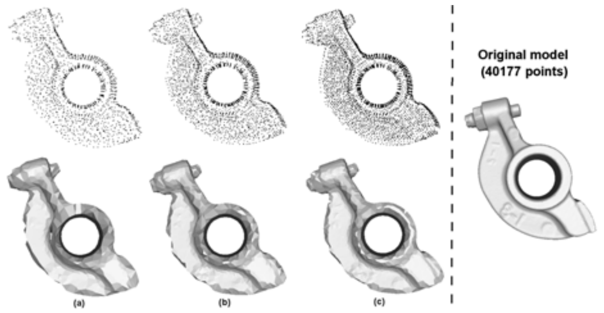

Figure 3. Uniformly simplified point sets of a genus-1 object and the renderings of their mesh reconstructions.
(a) 97.5% simplified ($\rho = 6.00$).
(b) 95.0% simplified ($\rho = 3.00$).
(c) 90.0% simplified ($\rho = 1.50$).

To demonstrate the use of our algorithm for feature-sensitive simplification, we adaptively simplify a point set sampled from the surface of the "Isis" model shown in figure 5(d). For simplicity, we consider the case of curvature-sensitive simplification only. We locally estimate the surface curvature on-the-fly across the input point cloud by, firstly, querying the Voronoi diagram generated by our simplification algorithm to obtain the set N_{p_i} of k nearest neighbours of the input point p_i to be weighted. Secondly, following the determination of the centroid of N_{p_i}, the covariance matrix $C_{N_{p_i}}$ of the points N_{p_i} around their centroid is computed. Eigenanalysis of the symmetric, positive semi-definite matrix $C_{N_{p_i}}$ may then be used to obtain a local curvature approximation. We follow Pauly et al. [9] by considering the ratio of the eigenvalue corresponding to the eigenvector in direction of the local surface normal to the sum of all eigenvalues as weight w_i of p_i. That is, $\tilde{F}(p_i) = w_i$. Figure 4(a) illustrates the feature-sensitive effect of these weights on the distribution of the resulting point set. Points are relatively more strongly concentrated in regions of change in curvature. As indicated in figure 4(b), any excessive irregularity of the resulting point sets undermining any meaningful further processing may be avoided by strengthening the density condition ρ. Any reduction in the value of ρ increases uniformity at the expense of adaptivity. Figure 5 shows the quality of the renderings supported by the point sets produced by curvature-sensitive, high-degree simplification.

Computational efficiency. In section 2.2, we determined our algorithm's worst case complexity of $O(N \log N)$, N denoting the number of grid points in Ω_r. Figure 6 presents uniform simplification execution times obtained experimentally. Figures 6(a) and 6(b) indicate that in practice the algorithm benefits from its coarse-to-fine nature in form of rather favourable execution time functions. Its performance is only weakly affected by substantial increases in either input model or target model size.

Point set distribution. For purposes of further processing of a simplified point set such as surface reconstruction or texture synthesis, its distribution is frequently re-

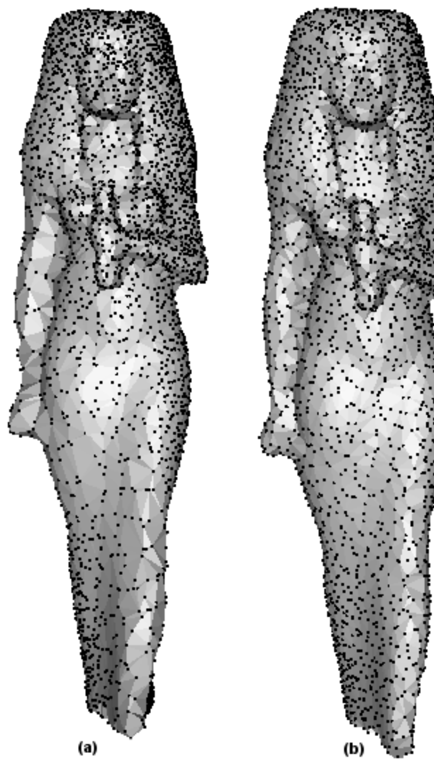

Figure 4. Feature-sensitively distributed point sets generated by adaptive simplification using local curvature estimation over rendering of corresponding mesh reconstructions. (a) $\rho = 8.00$ (b) $\rho = 5.50$.

quired to meet a uniform density requirement. As illustrated by figures 1 and 4, the algorithm presented here meets this requirement by taking into account a user-controlled density parameter ρ. The desired degree of uniformity is enforced irrespective of the degree of uniformity of the input point set. Due to the surface Voronoi diagram covering both over- and undersampled regions of the input point cloud, undersampled regions can be upsampled automatically by adjusting ρ as required.

Memory efficiency. The algorithm executes in-core using a grid data structure holding the offset band Ω_r and both a min- and max-heap. The memory requirements of the grid data structure depend on the size of the input point cloud and the radius r of the balls centred at the input points and used to determine the grid points to be included in Ω_r. The radius r may be allowed to vary adaptively alongside the grid density. As an approximate upper limit, the memory requirement follows $c * (max_i a_i)^3$, where a_i denotes the length of the point cloud's bounding box in the ith direction with c representing a small constant varying proportionally with the grid density.

The single min-heap is used to propagate multiple fronts simultaneously. Since, as part of the Fast Marching method, these fronts are only propagated in the direction of increasing distance, their size is substantially smaller than the size of the offset band at any time. The max-heap is used to hold the farthest point candidates. Its memory requirements therefore vary with the magnitude of the density parameter and the target model size respectively, which will generally be a fraction of the input model size.

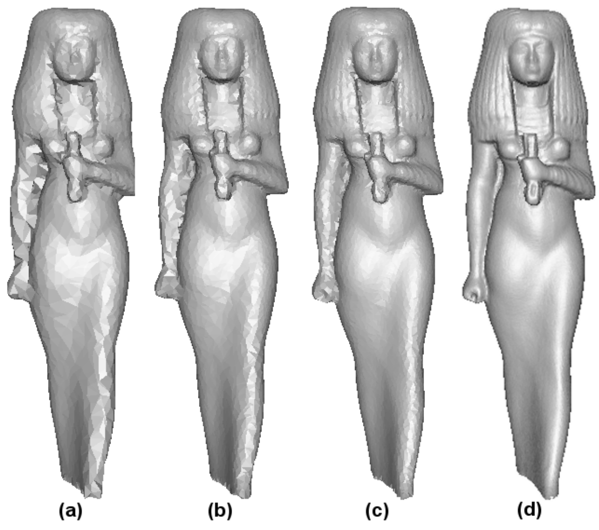

Figure 5. Renderings of meshes reconstructed from point sets generated by adaptive simplification using local curvature estimation.
(a) 97.5% simplified ($\rho = 8.50$).
(b) 95.0% simplified ($\rho = 4.30$).
(c) 90.0% simplified ($\rho = 2.10$).
(d) Original model (187644 points).

Approximation error. Qualitatively, as indicated by figures 2, 3 and 5, even for relatively small target model sizes, both the uniformly and feature-sensitively generated point sets allow for visually appealing reconstructions. For a more objective, quantitative evaluation of the extent of the geometric error introduced by the simplification, we are working on an automatic analysis tool which exploits the availability of the dual of the surface Voronoi diagram, i.e., the Delaunay triangulation of the point sets, to compute the distance between the surfaces represented by the input and output point sets.

4 Conclusion

We presented a new point cloud simplification algorithm with user-controlled density guarantee. The algorithm is computationally and memory efficient, easy to implement and requires no intermediate or prior surface reconstruction. Uniform point cloud simplification using our algorithm allows for high-quality further processing without the need for any prior resampling. Feature-sensitive simplification may be driven by any combination of point weights including colour differences and changes in curvature. The coarse-to-fine nature of the algorithm naturally supports the generation of progressive and multiresolution representations of the input point cloud.

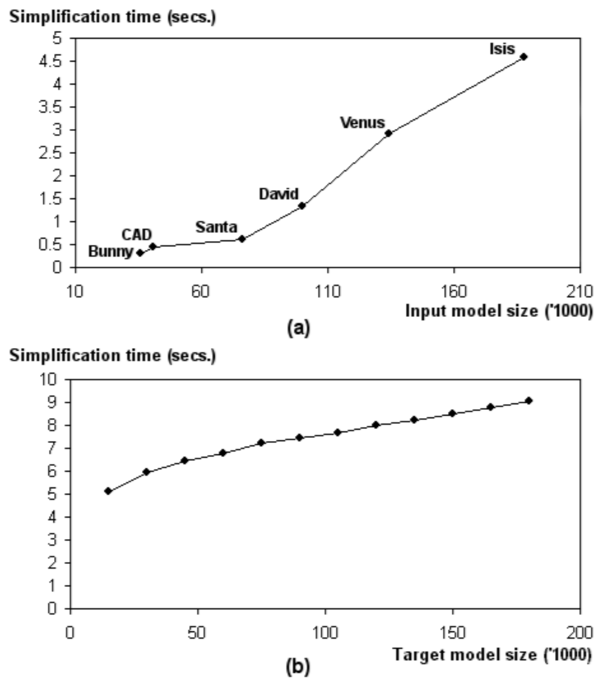

Figure 6. Simplification times in seconds on a Pentium 3, 700 MHz, Windows machine with 512MB main memory.
(a) Execution time as function of input model size (95% simplification of input point sets).
(b) Execution time as function of target model size.

We are currently working on an automatic tool for the quantitative analysis of the approximation error introduced by the algorithm. We are also interested in incorporating a method for the detection of noisy measurements in the input point set so that the need for the smoothing of point clouds produced by typical 3D surface acquisition devices becomes obsolete. We would further like to extend the algorithm to support (partial) out-of-core processing. Finally, we are investigating the generation of selectively-refined multiresolution representations of input point sets using our simplification algorithm.

Acknowledgements

We would like to thank Malcolm Sabin, Michael Blain, Mark Grundland and Marc Cardle for numerous helpful discussions. The point sets were obtained from the Cyberware and Stanford 3D Scanning Repository web sites.

References

[1] C. Moenning and N. A. Dodgson. Fast Marching farthest point sampling for implicit surfaces and point clouds. *Computer Laboratory Technical Report No. 565*, University of Cambridge, UK, 2003.

[2] L. Linsen. Point cloud representation. *CS Technical Report 2001-3*, Universität Karlsruhe, Germany, 2001.

[3] M. Pauly, L. Kobbelt and M. Gross. Multiresolution Modeling of Point-Sampled Geometry. *CS Technical Report #378*, ETH Zürich, Switzerland, 2002.

[4] H. Pfister, M. Zwicker, J. van Baar and M. Gross. Surfels: Surface Elements as Rendering Primitives. *SIGGRAPH '00, Annual Conf. Series*, New Orleans, USA, 2000, 335–342.

[5] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and T. Silva. Point Set Surfaces. *Proc. 12th IEEE Visualization Conf.*, San Diego, USA, 2001, 21–28.

[6] T. K. Dey, J. Giesen and J. Hudson. Decimating Samples for Mesh Simplification. *Proc. 13th Canadian Conference on Computational Geometry*, Waterloo, Canada, 2001, 85–88.

[7] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th Annual ACM Symposium on Computational Geometry*, Hong Kong, 2000, 213–222.

[8] J.-D. Boissonnat and F. Cazals. Coarse-to-fine surface simplification with geometric guarantees. *EUROGRAPHICS '01, Conf. Proc.*, Manchester, UK, 2001, 490–499.

[9] M. Pauly, M. Gross and L. P. Kobbelt. Efficient Simplification of Point-Sampled Surfaces. *Proc. 13th IEEE Visualization Conf.*, Boston, USA, 2002.

[10] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38, 1985, 293–306.

[11] Y. Eldar, M. Lindenbaum, M. Porat and Y. Y. Zeevi. The Farthest Point Strategy for Progressive Image Sampling. *IEEE Trans. on Image Processing*, 6(9), 1997, 1305–1315.

[12] A. Okabe, B. Boots and K. Sugihara. *Spatial Tessellations*. 2nd ed. (Chichester-UK: John Wiley & Sons, 2000).

[13] F. Mémoli and G. Sapiro. Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces. *Journal of Computational Physics*, 173(1), 2001, 764–795.

[14] F. Mémoli and G. Sapiro. Distance Functions and Geodesics on Point Clouds. *Technical Report 1902, IMA*, University of Minnesota, USA, 2002.

[15] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. 2nd ed. (Cambridge-UK: Cambridge University Press, 1999).